MỘT ĐỀ XUẤT CẢI TIẾN CHO KHAI THÁC K MẪU TUẦN TỰ

PHẠM THỊ THIẾT, PHẠM QUẢNG TRI, VÕ QUANG HOÀNG KHANG, VÕ ĐĂNG KHOA, HUỲNH TƯỜNG NGUYÊN

Khoa Công nghệ thông tin, Đại học Công nghiệp Thành phố Hồ Chí Minh

Tác giả liên hệ: phamthithiet@iuh.edu.vn

DOIs: https://www.doi.org/10.46242/jstiuh.v74i2.5001

Tóm tắt. Bài toán tìm ra k mẫu tuần tự trên cơ sở dữ liệu (CSDL) chuỗi tuần tự đã được nghiên cứu và đề xuất để giải quyết vấn đề làm thế nào người dùng có thể lựa chọn một ngưỡng hỗ trợ tối thiểu để tìm ra số lượng k mẫu người dùng mong muốn trong bài toán khai thác chuỗi tuần tự. Giải pháp này được đánh giá cao vì chi phí thực hiện thấp hơn nhiều so với bài toán khai thác chuỗi tuần tự truyền thống. Tuy nhiên hiện tại, quá trình tìm các mẫu tuần tự trong bài toán tìm k mẫu tuần tự đều được xây dựng theo hướng tiếp cận phát triển mẫu và CSDL chiếu nên tốn rất nhiều chi phí cho việc thực hiện các phép chiếu. Để giảm thời gian thực thi, bài báo này đề xuất dùng mã hóa khối nguyên tố để biểu diễn thông tin các ứng viên và tính toán độ hỗ trợ của ứng viên. Kết quả thực nghiệm cho thấy phương pháp đề xuất có thời gian thực thi tốt hơn so với phương pháp chiếu trên CSDL chiếu.

Từ khóa. mẫu tuần tự, k mẫu tuần tự, mã hóa khối nguyên tố.

1 GIỚI THIỆU

Tìm ra những tri thức, thông tin có giá trị từ khối lượng dữ liệu vô cùng lớn được hình thành từ nhiều loại ứng dụng khác nhau nhất là các ứng dụng trên internet để phục vụ cho cuộc sống con người, là công việc quan trọng của lĩnh vực khai thác dữ liệu nói chung, cũng như lĩnh vực khai thác mẫu tuần tự nói riêng. Khai thác mẫu tuần tự trên CSDL chuỗi tuần tự là một phần quan trọng trong lãnh khai thác dữ liệu và được ứng dụng rộng rãi trong nhiều lĩnh vực như y khoa, giáo dục đào tạo, kinh tế, khoa học, xã hội… Do vậy, đã có rất nhiều nghiên cứu được đề xuất đối với lĩnh vực này [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Khai thác mẫu tuần tự là tìm tất cả các chuỗi con có độ phổ biến lớn hơn ngưỡng hỗ trợ tối thiểu do người dùng định nghĩa (minsup), nghĩa là chúng xuất hiện nhiền hơn minsup lần trong CSDL chuỗi. Tuy nhiên, theo hướng tiếp cận này thì vấn đề là làm cách nào người sử dụng có thể chọn được giá trị minsup sao cho phù hợp để tìm ra đúng số lượng mẫu mà người dùng mong muốn. Điều này là khó khăn và rất tốn thời gian, nếu chọn giá trị minsup không phù hợp, thuật toán có thể tạo ra quá ít số lượng mẫu có ý nghĩa hay quá nhiều mẫu vô nghĩa. Một vấn đề quan trọng được đặt ra là làm thế nào để có thể tìm ra được đúng số lượng mẫu người dùng mong muốn mà không cần phải tính toán để xác định minsup là bao nhiêu thì tốt nhất. Để giải quyết vấn đề này, các nhà nghiên cứu gần đây đã đề xuất giải pháp khai thác k mẫu tuần tự với k là số mẫu tuần tự có độ hỗ trợ cao nhất được tìm thấy và được mong đợi bởi người dùng. Giải pháp này được đánh giá cao vì chi phí thực hiện thấp hơn nhiều so với các phương pháp khai thác mẫu tuần tự truyền thống. Giải pháp này không chỉ được áp dụng cho việc khai thác k mẫu tuần tự [11, 13], khai thác k mẫu tuần tự đóng như TSP [13], TKCS [14], mà còn hiệu quả trong nhiều bài toán khác trong lĩnh vực khai thác mẫu như: khai thác k itemset phổ biến [15, 16], khai thác k luật kết hợp [17], khai thác k luật tuần tự [18].

Thuật toán TSP [13] được Tzvetkov, Yan và Han đề xuất để khai thác top-k mẫu tuần tự phổ biến mà không yêu cầu người dùng một ngưỡng hỗ trợ tối thiểu minsup. Thuật toán được xây dựng theo hướng tiếp cận phát triển mẫu dựa trên thuật toán PrefixSpan [7] để tìm ra các mẫu tuần tự. Tuy nhiên, thuật toán TSP phải thực hiện việc duyệt và chiếu trên CSDL chiếu nhiều lần nên tốn chi phí rất cao đặc biệt là đối với CSDL dày.

Thuật toán TKS [11], được Fournier-Viger đề xuất bằng cách sử dụng thuật toán SPAM [2] để tạo ứng viên ban đầu sau đó tiến hành tìm và mở rộng các mẫu tuần tự. Thông tin của các mẫu ứng viên trong thuật toán được biểu diễn dưới dạng bit vec-tơ có kích thước cố định dẫn đến tiêu tốn rất nhiều thời gian để thực thi các phép giao bit vec-tơ.

Khái niệm mã hóa khối nguyên tố, là nền tảng của lý thuyết phân tích thừa số nguyên tố, được Karam Gouda và các cộng sự đề xuất sử dụng trong thuật toán PRISM (Prime-Encoding Based Sequence Mining) [4] để khai thác các mẫu tuần tự. Các kết quả thực nghiệm của PRISM đã được thực hiện, so sánh với thuật

toán PrefixSpan và SPAM cho thấy thuật toán PRISM đạt kết quả tốt nhất về tốc độ cũng như hiệu quả sử dụng bộ nhớ [4]. Thuật toán SPAM đứng thứ hai về tốc độ thực thi nhưng đối với CSDL lớn, thuật toán SPAM thất bại vì không đủ dung lượng bộ nhớ. Trong khi đó, PrefixSpan vẫn cho ra kết quả nhưng tốc độ thực thi chậm hơn so với PRISM.

Để nâng cao hiệu quả của bài toán khai thác k mẫu tuần tự, bài báo tập trung nghiên cứu và đề xuất thuật toán cho bài toán khai thác k mẫu tuần tự bằng cách sử dụng phương pháp mã hóa khối nguyên tố [4] để biểu diễn thông tin của các mẫu ứng viên. Dựa trên cách tiếp cận này, độ hỗ trợ của các mẫu ứng viên được tính toán trực tiếp từ khối mã hóa một cách nhanh chóng trong quá trình khai thác mẫu tuần tự. Cụ thể, bài báo bao gồm những đóng góp chính như sau:

- Sử dụng các khối mã hóa liên tiếp gồm các khối bit vec-tơ để biểu diễn các mẫu ứng viên. Khối mã hóa đầy đủ của một mẫu ứng viên bao gồm khối mã hóa vị trí và khối mã hóa chuỗi. Với mỗi khối mã hóa chuỗi có một chỉ số (position offsets) dùng để thu gọn khối mã hóa cũng như cho biết các khối vị trí nào là khác rỗng và bắt đầu từ vị trí nào trong dãy khối mã hóa vị trí. Đây cũng là cách nhằm loại bỏ đi những khối rỗng (bit rỗng) trong dãy mã hóa khối nguyên tố và tối ưu tài nguyên lưu trữ.
- Dựa vào mã hóa khối nguyên tố, việc tính toán sẽ nhanh hơn tại mỗi bước xử lý vì chủ yếu là thực hiện phép tính ước chung lớn nhất của các cặp số nguyên tố trong dãy khối mã hóa. Điều này giảm thời gian tính toán độ hỗ trợ cho từng ứng viên trong quá trình khai thác mẫu tuần tự.
- Đề xuất và hiện thực thuật toán đề xuất cho bài toán khai thác k mẫu tuần tự bằng cách sử dụng phương pháp mã hóa khối nguyên tố.
- So sánh, đánh giá kết quả thực nghiệm với thuật toán TSP [13] trên các CSDL khác nhau, đặc biệt trên CSDL dày và lớn.

Phần còn lại của bài báo được thiết kế như sau: Mục 2 trình bày một vài nghiên cứu liên quan đến bài toán khai thác k mẫu tuần tự, mẫu tuần tự đóng. Mục 3 mô tả các định nghĩa bài toán liên quan đến bài báo. Phương pháp bài báo đề xuất được đề cập trong mục 4. Các kết quả thực nghiệm được trình bày trong mục 5, kết luận và hướng phát triển được đề cập trong mục 6.

2 CÁC NGHIÊN CỨU LIÊN QUAN

Cho đến nay, nhiều nghiên cứu đã được đề xuất để giải quyết bài toán khai thác mẫu tuần tự trên CSDL chuỗi tuần tự như AprioriAll [19], GSP [9], SPADE [10], SPAM [2], Freespan [5], PrefixSpan [7], PRISM [4], MCM-SPADE [12] …. Tuy nhiên, các thuật toán trên đều yêu cầu một ngưỡng hỗ trợ tối thiểu minsup do người dùng định nghĩa. Trong thực tế, người sử dụng rất khó để chọn minsup đầu vào sao cho phù hợp để sinh tạo ra đúng số lượng mẫu mà người dùng mong muốn. Bởi vì, tùy thuộc vào minsup, thuật toán có thể trở nên rất chậm và tạo ra một số lượng rất lớn các kết quả không có giá trị hoặc quá ít kết quả hoặc không có kết quả nào, bỏ qua các thông tin có ích. Vấn đề này rất quan trọng bởi vì trong thực tế, tài nguyên về thời gian và không gian lưu trữ là có giới hạn để phân tích các kết quả và việc điều chỉnh thông số minsup sao cho phù hợp là rất tốn thời gian của người dùng. Để giải quyết vấn đề này, các thuật toán khai thác theo hướng k mẫu tuần tự phổ biến đã được nghiên cứu và đề xuất, với k là số mẫu có độ hỗ trợ cao nhất mà người dùng mong muốn thuật toán tìm kiếm và trả về mà không cần tính toán để xác định minsup là bao nhiêu. Đại diện cho hướng tiếp cận mới này, có các thuật toán sau như TSP [13], TKS [11]. TSP được Tzvetkov, Yan và cộng sự đề xuất để khai thác top-k mẫu tuần tự phổ biến mà không yêu cầu người dùng cung cấp giá trị minsup. Ý tưởng chính của thuật toán TSP là bắt đầu với minsup là 0, tiếp theo minsup được tăng dần trong quá trình xử lý và sau đó sử dụng minsup đã tăng để giảm không gian tìm kiếm. Nghĩa là ngay khi có k mẫu tuần tự được tìm thấy thì minsup sẽ được gán cho một giá trị là độ phổ biến của mẫu tuần tự có độ phổ biến nhỏ nhất trong k mẫu tìm được. Và như thế, minsup sẽ tiếp tục gia tăng trong suốt quá trình khai thác mẫu của thuật toán. Thuật toán TSP được xây dựng dựa trên cách tiếp cận mở rộng mẫu của thuật toán PrefixSpan [7], nghĩa là thuật toán thực hiện duyệt cơ sở dữ liệu để tìm các mẫu chứa duy nhất một phần tử. Sau đó, nó thực hiện đệ quy để mở rộng các mẫu và tìm các mẫu tuần tự bằng cách chiếu và quét trên cơ sở dữ liệu chiếu của nó. Ưu điểm chính của cách tiếp cận dựa trên phép chiếu được sử dụng là nó chỉ xem xét chuỗi con có tiền tố và chỉ chiếu chuỗi con hậu tố tương ứng của nó vào cơ sở dữ liệu chiếu mà không thực hiện việc kiểm tra khi phát sinh mẫu. Tuy nhiên, thuật toán TSP sử dụng phương pháp đệ quy để chiếu duyệt và chiếu trên CSDL chiếu nhiều lần nên rất tốn chi phí, đặc biệt với CSDL dày đặc

do có rất nhiều phép chiếu được thực hiện.

Thuật toán TKS [11] được đề xuất bởi Fournier-Viger và các cộng sự. Thuật toán sử dụng CSDL bitmap dọc để biểu diễn thông tin dữ liệu và sử dụng thủ tục SEARCH của thuật toán SPAM [2] tìm kiếm các mẫu tuần tự phổ biến cũng như tiến hành tìm và mở rộng các mẫu. Bên cạnh đó, thuật toán cũng áp dụng chiến lược để tăng hiệu quả khai thác k mẫu tuần tự bằng cách dùng danh sách L để lưu trữ các mẫu phổ biến đã tìm được cho tới hiện tại với điều kiện các mẫu trong L được sắp xếp theo độ hỗ trợ tăng dần. Khi có đúng k mẫu được tìm thấy, thuật toán tăng minsup lên bằng độ hỗ trợ của mẫu phổ biến có độ hỗ trợ thấp nhất trong L. Sau đó, mỗi khi một mẫu phổ biến được tìm thấy và thêm vào L thì loại bỏ các mẫu có độ hỗ trợ thấp nhất trong L cho tới khi L chỉ chứa đúng k mẫu, đồng thời tăng minsup lên bằng độ hỗ trợ của mẫu có độ hỗ trợ thấp nhất trong L. Thuật toán dừng khi không thể tạo được thêm một mẫu nào nữa.

Bài báo này đề xuất sử dụng phương pháp mã hóa khối nguyên tố [4] cho bài toán khai thác k mẫu tuần tự trong đó sử dụng số nguyên chính phương được phân tích thành các thừa số nguyên tố để biểu diễn thông tin của các ứng viên và tính toán độ hỗ trợ của các ứng viên trong quá trình khai thác mẫu tuần tự. Một tập gồm các số nguyên tố G={2, 3, 5, 11, 13, 17, 19} được dùng để phân hoạch thành các khối mã hóa liên tiếp cho các khối bit vec-tơ, ứng với các mẫu ứng viên là mã hóa theo chuỗi và mã hóa theo vị trí. Khối mã hóa đầy đủ của một mẫu ứng viên bao gồm khối mã hóa vị trí và khối mã hóa chuỗi; tương ứng với mỗi khối mã hóa chuỗi có một chỉ số (position offsets) dùng để thu gọn khối mã hóa cũng như cho biết các khối vị trí nào là khác rỗng và bắt đầu từ vị trí nào trong dãy khối mã hóa vị trí, đây cũng là cách nhằm loại bỏ đi những khối rỗng (bit rỗng) trong dãy mã hóa khối nguyên tố, tối ưu tài nguyên lưu trữ. Với mã hóa khối nguyên tố, khi khai thác dữ liệu, việc tính toán sẽ nhanh hơn tại mỗi bước xử lý vì chủ yếu là thực hiện phép tính ước chung lớn nhất của các cặp số nguyên tố trong dãy khối mã hóa.

3 CÁC ĐỊNH NGHĨA

Cho $I = \{i_1, i_2,\ldots,i_m\}$ là một tập gồm *m* phần tử (được gọi là item). Một itemset là tập khác rỗng và không có thứ tự của các item. Itemset *i* ký hiệu là $(i_1, i_2, \ldots, i_k)$ với mỗi i_j là một item. Một chuỗi (sequence) là một danh sách những itemset có thứ tự. Chuỗi *s* được ký hiệu là $\langle e_1, e_2,\ldots, e_n\rangle$ hoặc $\langle e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_n\rangle$ với mỗi e_i là một itemset, *n* là số lượng các itemset. Kích thước của một chuỗi là số lượng itemset có trong chuỗi đó. Chiều dài của một chuỗi là tổng số item có trong chuỗi, ký hiệu là $k = \sum_{j=1}^{n} e_j$. Chuỗi có chiều dài *k* còn được gọi là *k-sequence*. Ví dụ chuỗi s = ⟨(B)(AC)⟩ có 2 itemset gồm B và AC nên kích thước của chuỗi s là 2, số lượng item trong chuỗi s có 3 item gồm B, A, C nên chiều dài của chuỗi s là 3 và còn được gọi là 3-sequence

Chuỗi $\beta = \langle\beta_1, \beta_2, \ldots, \beta_m\rangle$ được gọi là chuỗi con của chuỗi $\alpha = \langle\alpha_1, \alpha_2,\ldots, \alpha_n\rangle$ hay α là chuỗi cha của β, ký hiệu $\beta \subseteq \alpha$, nếu tồn tại những số nguyên $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ sao cho $\beta_1 \subseteq \alpha_{j1}, \beta_2 \subseteq \alpha_{j2},\ldots, \beta_m \subseteq \alpha_{jn}$. CSDL chuỗi tuần tự bao gồm một tập hợp các chuỗi có dạng (*sid, s*), trong đó *sid* là định danh của chuỗi và *s* là tập các itemset. Một mẫu là một chuỗi con của một chuỗi dữ liệu. Mỗi một itemset trong một mẫu còn được gọi là một thành phần (element).

Cho CSDL chuỗi tuần tự *SD*, mỗi một chuỗi có một số định danh duy nhất. Độ hỗ trợ tuyệt đối của một mẫu *f* là tổng số chuỗi trong *SD* có chứa *f*, ký hiệu $sup_D(f) = |\{s_i \in SD \mid f \subseteq s_i\}|$. Độ hỗ trợ tương đối của *f* là tỉ lệ phần trăm các chuỗi trong *SD* chứa *f*. Ở đây, độ hỗ trợ tuyệt đối hoặc tương đối sẽ được sử dụng chuyển đổi qua lại, kí hiệu là *sup*(*f*). Cho trước *minsup,* một mẫu *f* được coi là mẫu phổ biến nếu độ hỗ trợ của nó lớn hơn hoặc bằng *minsup*: $sup(f) \geq minsup$, khi đó *f* được gọi là mẫu tuần tự.

Ví dụ: Cho CSDL chuỗi tuần tự *SD* như bảng 1 có tập các item phân biệt là { *C, D, F*} và *minsup* là 2. Xét chuỗi $s_1 = \langle(CD)CC(CD)CD\rangle$, chuỗi s_1 có 6 itemset là: (*CD*), *C*, *C*, (*CD*), *C*, *D* và có 8 item. Vậy s_1 có kích thước là 6 và có độ dài là 8. Trong chuỗi s_1, item *D* xuất hiện ba lần nhưng độ hỗ trợ của item *D* chỉ được tính là 1 đối với chuỗi s_1 đó. Chuỗi $q = \langle(CD)C\rangle$ là một chuỗi con của chuỗi s_1, vì vậy chuỗi con *q* còn được gọi là mẫu. Trong *SD*, chỉ có chuỗi s_1, s_2 và s_5 có chứa mẫu *q*, vậy độ hỗ trợ của mẫu *q* là 3. Vì $sup(q) > minsup$ nên *q* là một mẫu tuần tự.

Bảng 1 Cơ sở dữ liệu chuỗi *SD*

SID	Chuỗi
1	⟨(*CD*)*CC*(*CD*)*CD*⟩
2	⟨(*CD*)*CC*⟩
3	⟨*C*(*CD*)⟩
4	⟨*CCC*⟩
5	⟨(*CD*)(*CD*)(*CD*)(*D*) (*CF*)⟩

Bài toán khai thác k mẫu tuần tự là đi tìm k mẫu tuần tự trong CSDL mẫu tuần tự có độ hỗ trợ cao nhất với k là số lượng mẫu tuần tự mà người dùng mong muốn.

4 THUẬT TOÁN KHAI THÁC K MẪU TUẦN TỰ

Thuật toán đề xuất để khai thác k mẫu tuần tự gọi là MKSP (Mining K Sequential Patterns) được trình bày trong Hình 1.

Thuật toán này áp dụng phương pháp mã hóa khối nguyên tố để biểu diễn thông tin và giúp tính độ hỗ trợ của mẫu ứng viên một cách nhanh chóng. Thời gian thực hiện của thuật toán chủ yếu tập trung vào việc phát triển mẫu ứng viên từ tập *R* và thêm mẫu tuần tự mới vào tập *L* và *R* nên độ phức tạp về thời gian của thuật toán chủ yếu phụ thuộc vào kích thước của CSDL và số lượng mẫu. Do đó, độ phức tạp về thời gian có thể là *O(n.m)* (với *n* là số lượng chuỗi trong cơ sở dữ liệu và *m* là chiều dài trung bình của các chuỗi) hoặc cao hơn (tùy thuộc vào số lượng và kích thước của các mẫu). Thuật toán lặp đi lặp lại việc mở rộng các mẫu để tạo ra các mẫu dài hơn từ các mẫu ngắn hơn và độ phức tạp về thời gian của bước này phụ thuộc vào số lượng mẫu được tạo và kiểm tra độ hỗ trợ của mẫu. Do vậy độ phức tạp về thời gian của thuật toán là O(n^2 . m). Độ phức tạp về không gian của thuật toán *O(n)* hoặc cao hơn, nó chủ yếu phụ thuộc vào kích thước của tập *L* và *R*, cũng như các cấu trúc dữ liệu mà thuật toán thực tế sử dụng.

Hình 1: Thuật toán MKSP để tìm k mẫu tuần tự

Ví dụ: Với CSDL *SD* ở bảng 1 và k đầu vào là 2, thuật toán MKSP thực hiện được mô tả như sau:

- Đầu tiên, thuật toán khởi tạo *minsup* = 1; tập L, R được khởi tạo là rỗng.

```
Thuật toán MKSP:
Đầu vào: CSDL chuỗi SD, số nguyên k
Đầu ra: Tập k mẫu tuần tự
Phương pháp thực hiện:
    1. R := Ø. L := Ø. minsup := 1.
    2. Duyệt CSDL SD để tìm tất cả frequent 1-itemset và đưa vào tập R
    3. L← k mẫu trong R có độ hỗ trợ lớn nhất và gán
       minsup := độ hỗ trợ của mẫu trong L có độ hỗ trợ nhỏ nhất
    4. Loại bỏ tất cả các mẫu r ∈ R | sup(r) < minsup
    5. WHILE R != Ø DO
          Lấy p ∈ R có độ hỗ trợ lớn nhất để phát triển mẫu mới pnew và tính
    sup(pnew)dựa trên khối mã hóa nguyên tố
          IF (sup(Pnew) ≥ minSup) THEN
                L := L U {pnew}
                R := R U {pnew}
          WHILE |L| > k AND ∃s ∈ L | sup(s) = minsup
                Loại bỏ s ra khỏi L
          minsup := độ hỗ trợ của mẫu trong L có độ hỗ trợ nhỏ nhất
          Loại bỏ p ra khỏi R
          Loại bỏ tất cả các mẫu r ∈ R | sup(r) < minsup
      ENDWHILE
    return L;
```

- Sau đó, quét CSDL để tìm các mẫu đơn và đưa vào tập R, gồm {<C>: 5, <*D*>: 4, <F>: 1.

- Đưa 2 mẫu trong tập R có độ hỗ trợ lớn nhất vào tập L gồm {<C>: 5, < *D* >: 4}. Tăng *minsup*=4, sau khi loại bỏ các mẫu trong R có độ hỗ trợ nhỏ hơn *minsup*, tập R chứa các mẫu {< C>: 5, < *D* >: 4}.
- Lấy mẫu < *C* > có độ hỗ trợ cao nhất là 5 trong tập R ra để phát triển mẫu, lúc này thuật toán có những mẫu mới là < *CC* >: 5, < *CD* >: 3, bổ sung các mẫu mới < *CC* > vào tập R và tập L, thực hiện duy trì 2 mẫu (k=2) có độ hỗ trợ cao nhất trong tập L do đó loại bỏ mẫu < D >: 4 ra khỏi tập L và tăng *minsup*=5, khi đó, L chứa các mẫu gồm {< *C* >: 5 và < *CC* >: 5}, R chứa các mẫu {< *CC* >: 5, < D >: 4}.
- Với *minsup* = 5, thuật toán loại bỏ mẫu <*D*>: 4 ra khỏi tập R và lấy mẫu < *CC* >:5 trong tập R để phát triển mẫu. Tuy nhiên, không có mẫu mới nào được phát sinh với mẫu < *CC* >, do đó lúc này tập L vẫn tiếp tục được duy trì với 2 mẫu {< *C* >: 5, < *CC* >: 5}, đồng thời *minsup* = 5.
- Với *minsup* = 5, tại thời điểm này, tập R không còn chứa mẫu nào để lấy ra phát triển mẫu, do đó thuật toán dừng và trả về kết quả là 2 mẫu tuần tự là < *C* > và < *CC* > có độ hỗ trợ cao nhất bằng 5.

5 KẾT QUẢ THỰC NGHIỆM

Để đo hiệu suất của phương pháp đề xuất, các thực nghiệm được cài đặt trên máy tính Intel (R), Core (TM) i5-3320M CPU 2.6 GHz, 8Gb RAM, hệ điều hành Windows 10, ngôn ngữ lập trình Java. Các đánh giá thực nghiệm được hiện thực và so sánh với thuật toán TSP [13] trên bộ CSDL chuẩn gồm Mushroom, Leviathan, Sign và Kddcup99 được tải về từ địa chỉ trang web http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php. Đặc điểm của bộ dữ liệu Mushroom gồm 8124 chuỗi, 119 item phân biệt, số itemset trung bình trong mỗi chuỗi là 23; Bộ dữ liệu Leviathan gồm 5834 chuỗi, 9025 item phân biệt, số itemset trung bình trong mỗi chuỗi là 33; Bộ dữ liệu Sign gồm 730 chuỗi, 276 item phân biệt, số itemset trung bình trong mỗi chuỗi là 52. Bộ dữ liệu Kddcup99 gồm 20045 chuỗi, 50 item phân biệt, số itemset trung bình trong mỗi chuỗi là 16.

Các thực nghiệm của thuật toán MKSP và TSP được thực hiện ứng với mỗi giá trị *k* và kết quả thu được là giá trị trung bình sau 3 lần chạy. Thời gian thực thi đối với từng thuật toán được thể hiện trong bảng 2.

Bảng 2: Kết quả thực nghiệm ứng với *k* khác nhau

CSDL	**k**	**Thời gian thực thi (ms)**		
		MKSP	**TSP**	**Tỉ lệ (TSP/MKSP)**
Mushroom	1000	**5686**	10769	1.9
	2000	**9364**	18072.3	1.9
	3000	**14158**	23535.3	1.7
Leviathan	1000	**7702**	36563	4.7
	2000	**14070**	612167	43.5
	3000	**19234**	78251	40.7
Sign	1000	**2130**	3669	1.7
	2000	**3381**	6424	1.9
	3000	**4736**	11982	2.5
Kddcup99	1000	**15853**	52157	3.3
	2000	**30843**	Tràn bộ nhớ	
	3000	**47890**	Tràn bộ nhớ	

Với kết quả thực nghiệm được trình bày trong bảng 2 cho thấy thuật toán MKSP có thời gian thực thi nhanh hơn so với thuật toán TSP, đặc biệt với CSDL có kích thước lớn và nhiều item phân biệt thì thời gian của MKSP ít hơn rất nhiều so với TSP. Cụ thể như, với CSDL Mushroom có 119 item phân biệt, gồm 8124 chuỗi và itemset trung bình trên mỗi chuỗi là 23, khi người dùng muốn tìm k = 2000 mẫu tuần tự thì thời gian thực thi của TSP khai thác là 18072.3 ms trong khi đó thời gian xử lý của MKSP chỉ là 9364 ms. Hoặc với CSDL có kích thước lớn và nhiều item phân biệt như Leviathan có 5834 chuỗi, 9025 item phân biệt, với k = 3000 mẫu tuần tự thì thời gian của TSP là 78251 ms, trong khi đó MKSP là 19234 ms, thấp hơn khoảng 4 lần TSP. Bởi vì đối với CSDL có kích thước chuỗi lớn và nhiều item phân biệt như Leviathan so với Mushroom hoặc Sign thì việc xây dựng các CSDL chiếu để phục vụ cho mỗi bước đệ quy phát sinh mẫu của thuật toán TSP cũng tốn rất nhiều chi phí về thời gian xử lý hơn so với phương pháp mã hóa khối nguyên tố được sử dụng trong thuật toán MKSP. Đặc biệt đối với CSDL lớn như Kddcup99, thuật toán MKSP thực hiện được tất cả 3 trường hợp của tham số k, trong khi đó thuật toán TSP chỉ thực hiện được với trường hợp k=1000 còn 2 trường hợp còn lại thì thuật toán không thực hiện được bởi vì thuật toán thực hiện chiếu trên CSDL lớn sẽ dẫn đến việc tràn bộ nhớ do phải tạo ra quá nhiều CSDL chiếu.

Như vậy, qua kết quả thực nghiệm trên bộ các bộ CSDL khác nhau cho thấy thuật toán MKSP bài báo đề xuất có hiệu suất tốt hơn nhiều so với thuật toán TSP về thời gian khai thác; đặc biệt, MKSP chiếm ưu thế trên CSDL có kích thước mỗi chuỗi lớn và nhiều item phân biệt. Bởi vì thuật toán TSP tốn rất nhiều chi phí về thời gian xây dựng các CSDL chiếu để phục vụ cho mỗi bước đệ quy phát sinh mẫu, đặc biệt khi tăng giá trị k thì TSP không thể hoàn thành việc khai thác mẫu được do thiếu tài nguyên bộ nhớ trong khi đó thuật toán MKSP sử dụng phương pháp mã hóa khối nguyên tố để biểu diễn thông tin của các ứng viên làm tăng hiệu suất tính toán tại mỗi bước xử lý vì chủ yếu là thực hiện phép tính ước chung lớn nhất của các cặp số nguyên tố trong dãy khối mã hóa.

6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Bài báo đã đề xuất sử dụng phương pháp mã hóa khối nguyên tố [4] để góp phần làm tăng tốc độ khai thác của bài toán khai thác k mẫu tuần tự từ CSDL chuỗi. Thuật toán đề xuất đã được cài đặt thực nghiệm và thực hiện đánh giá thời gian thực thi với thuật toán TSP trên bộ CSDL chuẩn được sử dụng rỗng rãi trong cộng đồng nghiên cứu khai thác dữ liệu chuỗi. Kết quả thực nghiệm cho thấy thuật toán đề xuất MKSP có thời gian thực thi nhanh hơn so với thuật toán TSP.

Khám thác tri thức thông qua các mẫu tuần tự trong CSDL chuỗi là rất hữu ích và cấp thiết. Phương pháp đề xuất trong bài báo đã góp phần làm tăng tốc độ khai thác k mẫu tuần tự, hướng tiếp theo sẽ nghiên cứu, kiểm chứng và đánh giá hiệu quả về việc sử dụng tài nguyên bộ nhớ của phương pháp đề xuất và đảm bảo hoạt động tốt trên nhiều bộ CSDL khác nhau cũng như sẽ mở rộng đánh giá hiệu suất thực nghiệm của phương pháp đề xuất này với các thuật toán khác của bài toán khai thác top-k mẫu tuần tự, cụ thể là thuật toán TKS[11].

TÀI LIỆU THAM KHẢO

[1] R. Agrawal, R. Srikant. Mining Sequential Patterns. In: *Proceedings of the 11th Conference on Data Engineering (ICDE'95)*, pp. 3-14, 1995.

[2] J. Ayres, J.E. Gehrke, T. Yiu, J. Flannick. Sequential Pattern Mining using a Bitmap Representaion. In: *Proceedings of the Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 429-435, 2002.

[3] R. Bharathi, A. Noble Mary Juliet, M. L. Valarmathi. Mining sequential pattern for online shopping recommendation system using prefixspan algorithm. In: *Proceedings of the AIP Conference on Contemporary innovations in engineering and management*, vol. 2587, no. 1, pp. 120004, 2023.

[4] K. Gouda, M. Hassaan, M.J. Zaki. Prism: A Primal-Encoding Approach for Frequent Sequence Mining. *Journal of Computer and System Sciences*, vol. 76, no. 1, pp. 88-102, 2010.

[5] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 355-359, 2000.

[6] F. Masseglia, F. Cathala, P. Poncelet. The PSP Approach for Mining Sequential Patterns. In: *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pp. 176-184, 1998.

[7] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. In: *IEEE Trans. Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1424-1440, 2004.
[8] J. Pinaire, E. Chabert, J. Azé, S. Bringay, P. Landais. Sequential Pattern Mining to Predict Medical In-Hospital Mortality from Administrative Data: Application to Acute Coronary Syndrome. *Journal of Healthcare Engineering*, vol. 2021, pp. 1-12, 2021.
[9] R. Srikant, R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In: *Proceedings of the fifth International Conference on Extending Database Technology*, pp. 3-17, 1996.
[10] M.J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning 42*, pp. 31-60, 2001.
[11] P. Fournier-Viger, A. Gomariz, T. Gueniche, E. Mwamikazi, R. Thomas. TKS: Efficient Mining of Top-K Sequential Patterns. In: *Proceedings of the ninth International Conference on Advanced Data Mining and Applications (ADMA 2013)*, Part I, Springer LNAI 8346, pp. 109-120, 2013.
[12] B. Huynh, C. Trinh, H. Huynh, T.T. Van, B. Vo, and V. Snasel. An efficient approach for mining sequential patterns using multiple threads on very large databases. *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 242–251, 2018.
[13] P. Tzvetkov. "TSP: Mining Top-k Closed Sequential Patterns". *Knowledge and Information Systems*, vol. 7, no. 4, pp. 438-457, 2005.
[14] T. -T. Pham, T. Do, A. Nguyen, B. Vo and T. -P. Hong, An Efficient Method for Mining Top-K Closed Sequential Patterns. *IEEE Access*, vol. 8, pp. 118156-118163, 2020.
[15] C. Kun-Ta. Mining Top-k Frequent Patterns in the Presence of the Memory Constraint. *VLDB Journal*, vol. 17, no. 5, pp. 1321-1344, 2008.
[16] T. Le, B. Vo, V.-N. Huynh, N. T. Nguyen, and S. W. Baik. Mining top-*k* frequent patterns from uncertain databases. *Applied Intelligence*, vol. 50, pp. 1487–1497, Jan. 2020.
[17] P. Fournier-Viger. Mining Top-K Association Rules. In: *Proceedings of the 25th Canadian Conference on Artificial Intelligence (AI 2012)*, Springer, pp. 61-73, 2012.
[18] P. Fournier-Viger, V. S. Tseng. Mining Top-K Sequential Rules. In: *Proceedings of the 7th International Conference on Advanced Data Mining and Applications (ADMA 2011)*, Springer LNAI 7121, pp.180-194, 2011.
[19] W. Wang, J. Yang. Mining Sequential Patterns From Large Data Sets. *233 Spring Street, New York, NY 10013, USA: Springer Science + Business Media, Inc*, pp. 1-3, 2005.

A PROPOSING FOR MINING K SEQUENTIAL PATTERNS

PHẠM THỊ THIẾT, PHẠM QUẢNG TRI, VÕ QUANG HOÀNG KHANG, VÕ ĐĂNG KHOA, HUỲNH TƯỜNG NGUYÊN

Faculty of Information Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City

Corresponding author: phamthithiet@iuh.edu.vn

Abstract. The problem of finding k sequential patterns from a sequence database has been researched and proposed to solve the problem of how users can choose a minimum support threshold to find the number k desired user patterns in the sequential pattern mining problem. This solution is highly appreciated because the implementation cost is much lower than the traditional sequential pattern mining problem. However, currently, the process of finding sequential patterns in the problem of finding k sequential patterns is built according to the pattern development and projection database approach, so it costs a lot of money to perform projections. To reduce execution time, this paper proposes to use prism block encoding method to represent pattern information and calculate pattern support. Experimental results show that the proposed method has better execution time than the projection method on projection database.

Keywords. sequential pattern, k sequential pattern, prism block encoding.

Ngày nhận bài: 13/3/2024
Ngày nhận đăng: 06/3/2025